Mathematically Reduced Chemical Reaction Mechanism Using Neural Networks

Technical Progress Report

Period ending: 09/30/2006

Prepared by

Nelson Butuk
Principal Investigator
Department of Mathematics
Prairie View A & M University
Prairie View Texas. 77446-4189

March, 2007

DOE Grant Number: DE-FG26-03NT-41913

Office of Sponsored Programs
Prairie View A & M University
P. O. Box 667
Prairie View Texas 77446-0667

Disclaimer

Abstract

This is an annual technical report for the work done over the last year (period ending 9/30/2005) on the project titled "Mathematically Reduced Chemical Reaction Mechanism Using Neural Networks." The aim of the project is to develop an efficient chemistry model for combustion simulations. The reduced chemistry model will be developed mathematically without the need of having extensive knowledge of the chemistry involved. To aid in the development of the model, Neural Networks (NN) will be used via a new network topology know as Non-linear Principal Components Analysis (NPCA).

We report on the significant development made in developing a truly meshfree computational fluid dynamics (CFD) flow solver to be coupled to NPCA. First, the procedure of obtaining nearly analytic accurate first order derivatives using the complex step method (CSM) is extended to include computation of accurate meshfree second order derivatives via a theorem described in this report. Next, boosted generalized regression neural network (BGRNN), described in our previous report is combined with CSM and used to obtain complete solution of a a hard to solve wave dominated sample second order partial differential equation (PDE): the cubic Schrodinger equation. The resulting algorithm is a significant improvement of the meshfree technique of smooth particle hydrodynamics method (SPH). It is suggested that the demonstrated meshfree technique be termed boosted smooth particle hydrodynamics method (BSPH). Some of the advantages of BSPH over other meshfree methods include; it is of higher order accuracy than SPH; compared to other meshfree methods, it is completely meshfree and does not require any background meshes; It does not involve any construction of shape function with their associated solution of possibly ill conditioned matrix equations; compared to some SPH techniques, no equation for the smoothing parameter is required; finally it is easy to program.

TABLE CONTENTS

Disclaimer	2
Abstract	3
Introduction	5
The Complex Step Method (CSM) of Derivatives	6
Theorem 1 for 2nd Derivatives	7
Stability of Time Stepping: Runge-Kutta Errors	10
New Error Perturbation Method:	12
Results and Discussions	13
Conclusion	24
References	17
Appendix	18

1. Complete paper: A New Meshfree Accurate Method for Solving the Time Dependent Schrodinger Equation: 8 pages
2. Complete FORTRAN Code: 8 pages
3. MSc. Thesis Supported by DOE: Ken Johnson: 5 pages

Introduction

This is an annual technical report for the work done over the last year (period ending 9/30/2006) on the project titled "Mathematically Reduced Chemical Reaction Mechanism Using Neural Networks." The aim of the project is to develop an efficient chemistry model for combustion simulations. The reduced chemistry model will be developed mathematically without the need of having extensive knowledge of the chemistry involved. To aid in the development of the model, Neural Networks (NN) will be used via a new network topology know as Non-linear Principal Components Analysis (NPCA).

Many Combustion systems are modeled by very high-dimensional systems of non-linear differential equations. These equations often exhibit solutions which are un-evenly distributed in phase-space, and which may exist as circles, tori or other manifolds. It is desirable to approximate these isolated regions of the phase-space by a mathematical model of lower dimension than the dimension of the original ambient space. We propose to develop NPCA to accomplish this task using NN.

Given a data set $X \in R^n$ NPCA determines a reduction mapping

$$G: X \rightarrow Y$$

where the set $Y \in R^m$ has reduced dimension $m < n$. Y is then said to be a reduction of X. The inverse mapping reproduces the original data X from the reduced data set Y as

$$H: Y \rightarrow X$$

Hence, NPCA is a composition of mappings which is the same as the identity mapping, i.e.

$$H \text{ o } G: X \rightarrow X$$

The NPCA is a NN topology with five layers, in which the input layer has the same number of nodes as the output layer. This allows the input values to be used as target output values of the network during training. The first part of the network approximates the mapping G. It contains the mapping layer. The middle layer is the bottle-neck layer consisting of the m nodes (i.e. desired reduced dimension m). The last part of the network implements the mapping H and contains the de-mapping layer. This layer takes the output of the middle layer and maps it onto the output layer. It is on the bottle-neck layer, that the reduced manifold of the chemistry is reconstructed.

The Objectives of this research are therefore:

1. Improve the training rate of the NPCA-NN algorithm developed previously.
2. Develop a method to determine optimum trajectory data of reaction mechanisms needed to use NPCA-NN.
3. Apply the NPCA-NN algorithm to the reduction of Dimethyl ether (DME) mechanism.
4. Couple the developed NPCA-NN model to the KIVA CFD code.
5. Test the CFD code on a few other simple sample mechanisms
6. Student Education in Computational Applied Mathematics

Continued in this report is work accomplished in developing a new mesh free approach to modify our in-house developed CFD code in partial fulfillment of objective 4. The advantage of developing our own CFD code, is that it will be easier to incorporate new mathematical models for complex physical systems such as reactive flows via developed NPCA-NN and models for other areas such as turbulence and multiphase flows.

This report begins by describing the complex step method (CSM) of obtaining nearly analytic accuracy, this is followed by a description on extending the method to computing second order derivatives, next is described a new algorithm for automatic step size control during time integration of deferential equations, then results of testing of the new methods described are presented. Finally in fulfillment of objective (6) above, a few pages of the masters thesis completed with the support of this project is attached in the appendix. The work reported here will be submitted to Texas A & M University Technology Office for review for possible intellectual property protection. This will be done as soon as the final report is submitted to DOE at the conclusion of this project on 5-31-2007.

The Complex Step Method (CSM) of Derivatives

In the last annual report [Butuk, 2006] details of the CSM has been described. In that report the method was used to compute accurate first order derivatives. It was shown that when combined with boosted generalized regression neural network (BGRNN) a robust function estimator that can be used for meshfree CFD was realized. The problem was that CFD computations require the estimation of not only first order derivatives but also second order derivatives. Currently in CFD, various finite difference approximations are used to estimate these derivatives. The problem for meshfree CFD, is that the estimates are not grid size independent and in most cases require structured grids to describe the geometry. This is where the CSM may become useful as the first order derivatives can be computed in a completely grid size independent manner. If the second order derivatives can be computed in a similar manner then the CSM will be a very useful tool for meshfree CFD. In this report, it will be described for the first time a new technique of computing second order derivatives using CSM to take advantage of its inherent meshfree character for first order derivatives. First a brief description of the CSM method is given.

To derive the first order derivative approximation of a non-linear function, $f(x)$, expand via a complex argument, x + ih i = √-1

$$f(x+ih) = f(x) + ihf'(x) - \frac{h^2 f'(x)}{2} - \frac{ih^3}{6} f'''(x) + \frac{h^4}{24} f^{iv}(x) + \ldots \quad (1)$$

Taking the imaginary parts,

$$\mathrm{Im}[f(x+ih)] = hf'(x) - \frac{h^3}{6} f'''(x)$$

Rearranging, obtain

$$f'(x) = \frac{\mathrm{Im}[f(x+ih)]}{h} + O(h^2) \quad (2)$$

To derive, the second order derivative, consider now the real component of (1) which is

$$\mathrm{Re}\left[\frac{h^2}{2} f''(x)\right] = f(x) - \mathrm{Re}\left[f(x+ih)\right] + \frac{h^4 f^{iv}(x)}{24}$$

Solving for the second order derivative we obtain

$$f''(x) = \frac{2}{h^2}\left\{ f(x) - \mathrm{Re}[f(x+ih)]\right\} + O(h^2) \tag{3}$$

As can be seen, unlike the first order derivative formula, this equation for second order derivatives suffers from subtractive cancellation errors and will not be step size independent. Kok-Lam et al. (2005), has indicated that the use of this equation to compute second order derivatives is worst than using a finite difference scheme. Because of this they have developed a new method which extends (3) in combination with Richardson extrapolation. Their work resulted in a more accurate formula for the derivatives but was still dependent on the step size, h chosen. Here, in this report is described a method that is completely step size independent for second order derivatives. The description of the method is via the following theorem:

Theorem 1:

Let $f(x)$ be a real valued function. If its complex extension $f(z)$ is analytic and can be expanded with a complex Taylor series, that is the expansion of $f(x + ih)$ where $i = \sqrt{-1}$ and $h << 1.0$ gives

$$\frac{\partial f}{\partial x} = \frac{\mathrm{Im}[f(x+ih)]}{h} + O(h^2)$$

then the second-order partial derivatives can be computed given a set of functional values at grids points $\left\{x_j\right\}_{j=1}^{n}$ by the step size h, independent (or meshfree) formulae

$$f''_{j-1} + 10 f''_j + f''_{j+1} = \frac{12\left[f_{j-1} - 2f_j + f_{j+1}\right]\left[f'_{j-1} + 4f'_j + f'_{j+1}\right]^2}{9\left[f_{j+1} - f_{j-1}\right]^2} + O(h^4) \qquad for \;\; j = 2...n-1$$

with the values of partial derivatives at the boundaries determined by a suitable low-order approximation that can also be made to be step size independent.

Proof:

The complete proof makes use of the Taylor series expansion method. Given a set of points x_j with functional values $f(x_j)$, the proof starts with 3 adjacent points x_{j-1} x_j and x_{j+1} uniformly spaced by distance h. The corresponding functional values are f_{j-1} f_j and f_{j+1}. Now with point j taken as the origin the Taylor's expansions for j-1 and j+1 are

$$f_{j-1} = f_j - hf'_j + \frac{h^2}{2} f''_j - \frac{h^3}{6} f'''_f + \frac{h^4}{24} f_j^{iv} - \frac{h^5}{120} f_j^{v} + \frac{h^6}{720} f_j^{vi} + ... \tag{4}$$

$$f_{j+1} = f_j + hf'_j + \frac{h^2}{2} f''_j + \frac{h^3}{6} f'''_f + \frac{h^4}{24} f_j^{iv} + \frac{h^5}{120} f_j^{v} + \frac{h^6}{720} f_j^{vi} + ... \tag{5}$$

To derive, finite difference formula for derivatives, the standard approach [Chung ,2002] is to construct a linear combination of Taylor's expansions and determine the expansion coefficients

by maximizing the accuracy based on a given stencil (or set of points). Hence to proceed, we construct a difference formulae for maximum accuracy based on a 3-point stencil (j-1 j and j+1). Given functional values f_{j-1} f_j and f_{j+1} construct a difference formula for f''_{j-1} f''_j and f''_{j+1} as a linear combination:

$$f''_j + a_o f_j + a_1 f_{j+1} + a_2 f_{j-1} + a_3 f''_{j+1} + a_4 f''_{j-1} = O(h^y) \tag{6}$$

The objective is to determine the coefficients to minimize the truncation error (RHS of equation 6) for h << 1.0 i.e. make y to be as high as possible. To be able to determine the coefficients a_o … a_4, using Taylor's expansions, we need in addition to equations (4) and (5) , Taylor's expansions for f''_{j-1} and f''_{j+1}

$$f''_{j-1} = f''_j - hf'''_j + \frac{h^2}{2} f^{iv}_j - \frac{h^3}{6} f^{v}_j + \frac{h^4}{24} f^{vi}_j + \ldots \tag{7}$$

$$f''_{j+1} = f''_j + hf'''_j + \frac{h^2}{2} f^{iv}_j + \frac{h^3}{6} f^{v}_j + \frac{h^4}{24} f^{vi}_j + \ldots \tag{8}$$

Next substitute the Taylor's expansions (4) , (5), (7), and (8) into (6) after re-arranging obtain

$$\begin{aligned}
&\left(a_o + a_1 + a_2\right) f_j + h\left(a_1 - a_2\right) f'_j \\
&+\left(1 + \frac{a_1 h^2}{2} + \frac{a_2 h^2}{2} + a_3 + a_4\right) f''_j \\
&+\left(\frac{a_1 h^3}{6} - \frac{a_2 h^3}{6} + a_3 h - a_4 h\right) f'''_j \\
&+\left(\frac{a_1 h^4}{24} + \frac{a_2 h^4}{24} + \frac{a_3 h^2}{2} + \frac{a_4 h^2}{2}\right) f^{iv}_j \\
&+\left(\frac{a_1 h^5}{120} - \frac{a_2 h^5}{120} + \frac{a_3 h^3}{6} - \frac{a_4 h^3}{6}\right) f^{v}_j \\
&+\left(\frac{a_1 h^6}{720} + \frac{a_2 h^6}{720} + \frac{a_3 h^4}{24} + \frac{a_4 h^4}{24}\right) f^{vi}_j + \ldots
\end{aligned} \tag{9}$$

To obtain the highest accuracy, we set as many of the low-order terms to zero as possible. Since there are five unknowns (a_o … a_4) , we will set the coefficients of the first five terms to zero obtaining

$$
\begin{aligned}
&a_o + a_1 + a_2 = 0 \\
&(a_1 - a_2)h = 0 \\
&1 + \frac{a_1 h^2}{2} + \frac{a_2 h^2}{2} + a_3 + a_4 = 0 \\
&\frac{a_1 h^3}{6} - \frac{a_2 h^3}{6} + a_3 h - a_4 h = 0 \\
&\frac{a_1 h^4}{24} + \frac{a_2 h^4}{24} + \frac{a_3 h^2}{2} + \frac{a_4 h^2}{2} = 0
\end{aligned} \tag{10}
$$

These five equations are solved with the aid of Maple Symbolic Algebra system to obtain

$$
a_o = \frac{24}{10h^2} \quad a_1 = a_2 = \frac{-12}{10h^2}
$$

$$
a_3 = a_4 = \frac{1}{10}
$$

The truncation error is obtained by substituting these values into the second last term of (9), when this is done this term also vanishes, so the values are substituted into the last term to obtain

$$
\frac{-12h^6}{7200h^2} - \frac{12h^6}{7200h^2} + \frac{h^4}{240} + \frac{h^4}{240}
$$

$$
= O\left(h^4\right)
$$

Hence the value of y in (6) is 4. Finally substituting into (6) the desired difference equation is obtained:

$$
f''_{j-1} + 10 f''_j + f''_{j+1} = \frac{12}{h^2}\left[f_{j-1} - 2 f_j + f_{j+1} \right] + O(h^4) \tag{11}
$$

This formula contains the grid spacing parameter, h and is therefore not meshfree. Our aim is to make it meshfree. The key innovation reported, is to use the fact that the complex step method (CSM) of obtaining first order partial derivatives as required by the above theorem, is very accurate and robust and in particular is meshfree or step size h independent. To use this fact, we will develop another difference formula for first order partial derivatives: Following the same approach as (6) write:

$$
f'_j + a_o f_j + a_1 f_{j+1} + a_2 f_{j-1} + a_3 f'_{j+1} + a_4 f'_{j-1} = O(h^y)
$$

To proceed we require the following Taylor's expansions for

$$
f'_{j-1} = f'_j - h f''_j + \frac{h^2}{2} f'''_j - \frac{h^3}{6} f^{iv}_j + \frac{h^4}{24} f^{v}_j + \ldots
$$

$$
f'_{j+1} = f'_j + h f''_j + \frac{h^2}{2} f'''_j + \frac{h^3}{6} f^{iv}_j + \frac{h^4}{24} f^{v}_j + \ldots
$$

The final difference equation obtained is:

$$f'_{j-1} + 4f'_j + f'_{j+1} = \frac{3(f_{j+1} - f_{j-1})}{h} + O(h^4) \tag{12}$$

Now solve (12) for grid parameter h and substitute into (11) to obtain

$$f''_{j-1} + 10f''_j + f''_{j+1} = \frac{12\left[f_{j-1} - 2f_j + f_{j+1}\right]\left[f'_{j-1} + 4f'_j + f'_{j+1}\right]^2}{9\left[f_{j+1} - f_{j-1}\right]^2} + O(h^4) \tag{13}$$

Completing the proof

Practical Implementation: Equation 13 forms a tridiagonal system of equations that is efficiently implemented via a standard tridiagonal solver in O(n) operations. One has to specify the partial derivatives at the boundaries j=1 and j=n. This is achieved via a low-order formula that can also be derived to be meshfree. In CFD various boundary specification techniques has been described by Poinsot and Lele, (1992).

Since this theorem was used in a PDE solver, it is necessary to describe the time stepping algorithm that was implemented when solving the PDE. This is accomplished next.

Stability of Time Stepping: Runge-Kutta Errors

Given a differential equation of the form

$$\frac{dy}{dt} = f(t, y) \tag{14}$$

A first order Tayor's integration formular is

$$\begin{aligned} y(t + \Delta t) &= y(t) + y'(t)\Delta t \\ &= y(t) + f(t, y)\Delta t \end{aligned}$$

Showing that an explicit solution at the next time step (t + Δt) can be obtained from the solution at the previous time, y(t). This is a first order Runge-Kutta (R-K) formula also known as Euler's method. Due to excessive truncation error, this method is rarely used in practice. To improve the error, a second-order R-K formula can be developed as:

$$\begin{aligned} K_o &= \Delta t f(t, y) \\ K_1 &= \Delta t f\left(t + \frac{\Delta t}{2}, y + \frac{K_o}{2}\right) \\ y(t + \Delta t) &= y(t) + K_1 \end{aligned} \tag{15}$$

Although more accurate than Euler's method, this second-order method is less popular and the more accurate 4^{th} –order R-K is widely used and is almost always the standard choice for the majority of ordinary differential equations (ODEs). This 4^{th} –order R-K formula is

$$
\begin{aligned}
K_o &= \Delta t f(t, y) \\
K_1 &= \Delta t f\left(t + \frac{\Delta t}{2}, y + \frac{K_o}{2}\right) \\
K_2 &= \Delta t f\left(t + \frac{\Delta t}{2}, y + \frac{K_1}{2}\right) \\
K_3 &= \Delta t f\left(t + \Delta t, y + K_2\right) \\
y(t + \Delta t) &= y(t) + \frac{K_o + 2K_1 + 2K_2 + K_3}{6}
\end{aligned} \tag{16}
$$

Consider the last equations in (15) and (16), these will be used to describe the method implemented for automatic error/stability control in the solution of a set of partial differential equations (PDEs) described below in this report and used to demonstrate the practical implementation of the meshfree method of computing derivatives described above. In this report, an adaptive method has been implemented which evaluates the truncation error at each time step and adjusts the value of Δt accordingly.

The aim of adaptive error control, is to ensure that the effects of local errors do not accumulate catastrophically; that is the global error should remain bounded and be minimized as time integration progresses. If the method of integration is unstable, the global error will increase exponentially eventually causing numerical overflow. Hence, our aim is to control the error to provide for stability. Stability is determined by three factors: the differential equation type, the method of solution chosen, and the value of the time step chosen Δt. Given an ODE and a method, Δt can be used to control stability. In a meshfree solver therefore, Δt should be the only parameter to effect stability and not the grid parameter, h (discussed above), i.e. grid spacing.
In adaptive R-K methods, what are known as embedded integration formulas are used [Press, et. al., 1992]. These formulas come in pairs. One formula has the integration order m, the other is of order m+1. The idea is to use both formulas to advance the solution from t to t+ Δt. Let $y_m(t + \Delta t)$ be the m^{th} -order solution and y_{m+1} be the $(m+1)^{th}$-order solution (e.g. equations 15 and 16 above), the truncation error of order m is then estimated as follows:

$$
E(\Delta t) = y_{m+1}(t + \Delta t) - y_m(t + \Delta t) \tag{17}
$$

Let Δt be represented by h for convenience (not to be confused with grid parameter h above). Hence once $y_m(t+h)$ has been computed, relatively small additional effort is required to compute $y_m(t + h)$. Note that when the number of ODEs is n, then E(h) is a vector with components $\{E_1 \ldots E_n\}$.

Local errors accumulate step by step leading to a global error after several steps. In a particular R-K method, if the term with the lowest power of h not included in the truncated Taylor's expansion (used to develop the method) involves h^{k+1} , then the global error at a particular time, t will be approximately proportional h^k. This fact helps us to develop ideas on how to control the error at each time step.

For step size control, we require that at each step, the local truncation error, E(h) should be bounded by ϵ , a desired upper bound on error. Hence

$$\|E(h)\| \leq \in$$

Note that, the notation ||E|| denotes infinity norm, which is the magnitude of the largest component

$$\|E\| = \max\{|E_1|,\ldots|E_n|\}$$

Since the local truncation error is based on 4^{th} –order R-K, it is of $O(h^5)$ i.e.

$$E(h) \approx ch^5 \qquad |h| << 1$$

If h_o is the trial step size used over the interval from t to t + h, and let h_1 denote the new step size. It can be shown that the step size can be computed as

$$h_1 = \left(\frac{\in}{\|E(h)\|}\right)^{\frac{1}{5}} h_o \tag{18}$$

Hence, if the estimated truncation error at each time step is large then the step size is reduced, if the error is small, the solution is accepted and the step size is increased. In practical implementation, it is more convenient to use relative error as a criteria for changing the time step. At each time step

$$\|E(h)\| \leq \max\{\|y_{m+1}\|, 1.0\} \in$$

The relative error is used except when the solution becomes small in the sense that

$$\|y_{m+1}\| < 1.0$$

Since equation (18) is an approximation, it is usual to incorporate a safety factor that is less than one in the update formula. The update formula implemented in this report is therefore

$$h_1 = 0.9\frac{\left[\max\{\|y_{m+1}\|, 1.0\} \in\right]^{\frac{1}{4}}}{\|E(h)\|} h_o \tag{19}$$

In equation (19) the exponent used is ¼ instead of 1/5 since this report implements the low storage 4^{th} –order R-K describe in [Lobo, 1997]. This scheme is described next.

An m-stage R-K scheme (with low storage requirements) can be derived as:

$$\begin{aligned}
&y_1 = y(x) + \alpha_1 h f(y)\\
&y_2 = y(x) + \alpha_2 h f(y_1)\\
&\ldots\\
&\ldots\\
&y_{m-1} = y(x) + \alpha_{m-1} h f(y_{m-2})\\
&y(x+h) = y_m = y(x) + \alpha_m h f(y_{m-1})
\end{aligned} \tag{20}$$

In this report m was chosen as 4 to be equivalent to the classical 4^{th} –order R-K. The estimate at the 3^{rd} stage was used as the first solution with the 4^{th} –stage solution providing the second solution: hence, obtaining the necessary pair of solutions for the embedded integration formula as described above in implementing automatic step size control.

New Error Perturbation Method:

One new idea implemented in this report and which appears to improve the solution accuracy by an order of magnitude (a change of residuals sum of squares (RSS) when solving the PDE to be described below decreased from 5.52 to 0.52) without changing the computational time is as follows: At each rejected time step let the errors be

$$\hat{\varepsilon}_i = E_i(h) / \left\| E(h) \right\|$$

Create a set of normalized errors

$$\tilde{\varepsilon}_i = \hat{\varepsilon}_i - \frac{1}{n} \sum_j \hat{\varepsilon}_j$$

with mean zero. These errors are then sampled randomly and added to the current solution at the beginning of the time step.

$$y_i(t) = y_i(t) + \tilde{\varepsilon}_i$$

After adding the random error the integration step is repeated. This method appears to be promising in providing stability and ensuring that the minimum time step is not encountered. More numerical experiments needs to be carried out on this method as it maybe related to the popular idea of adding artificial viscosity in standard CFD solvers.

Results and Discussions

This section will discuss the results demonstrating the above theorem when the complex step method (CSM) of derivatives is applied to three sample problems: computing first and second order derivatives of a 3^{rd} degree polynomial function via application of CSM directly, computing first and second order derivatives of a 3^{rd} degree polynomial function via approximation of function using data and boosted generalized regression neural network (BGRNN), and solving a second order partial differential equation (PDE) using CSM with R-K time stepping algorithm as described above.

Table 1, presents the results of applying the CSM directly to the following polynomial function

$$f(x) = 4 + 3x + 4x^2 + x^3 \tag{21}$$

The plot of this function is shown in Figure 1 below. The analytic first and second derivatives are

$$\begin{aligned} f'(x) &= 3 + 8x + 3x^2 \\ f''(x) &= 8 + 6x \end{aligned} \tag{22}$$

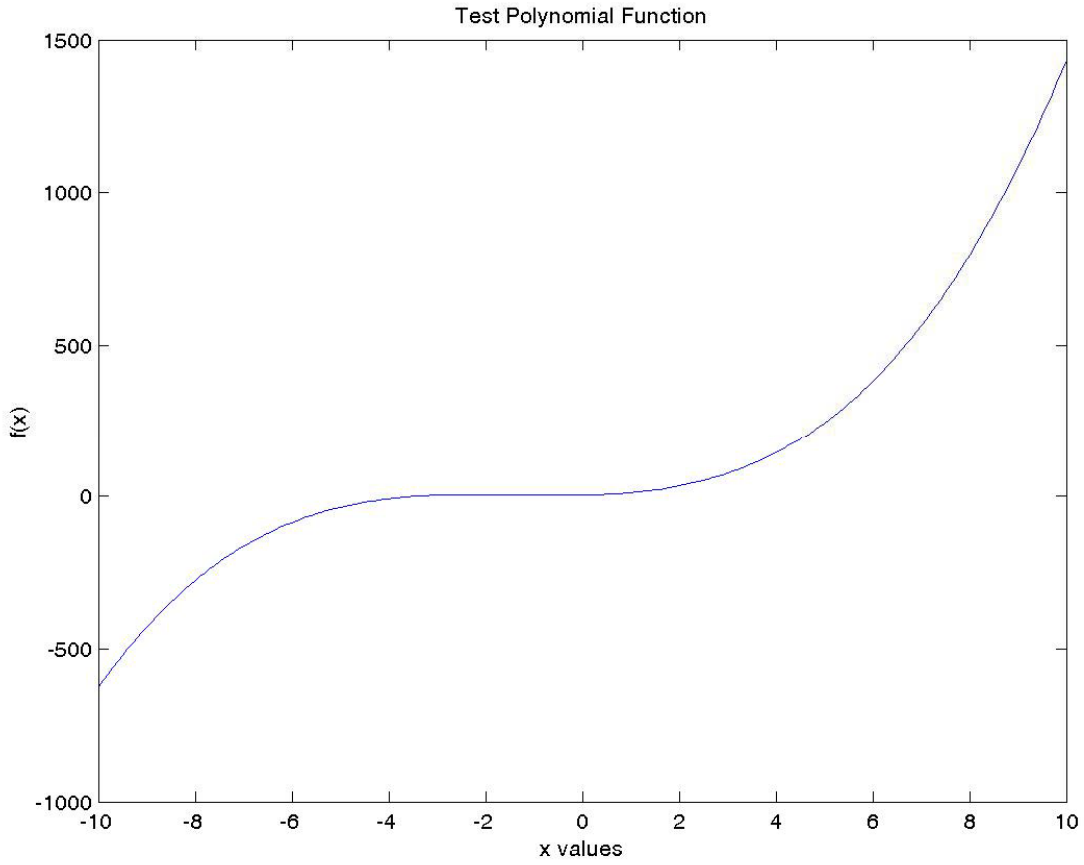

Figure 1. Polynomial Test Function of Equation (21)

In the table x ϵ [-10,10] , F are the function values, FDX are the function first-order analytic derivatives, FDXX are the second order analytic derivatives, and ERR-F, ERR-FDX, ERR-FDXX are relative percentage errors for estimating the function, first derivative, and second derivatives respectively using the CSM directly. The above theorem 1, was used to estimate the second order derivatives. As can be seen these results are highly encouraging as they indicate nearly analytic accuracy even for the second order derivatives, which is in agreement with the 4^{th} order accuracy of method indicated in above theorem. The relative percentage errors were computed as

$$error = \frac{analytic - estimate}{analytic} 100 \tag{23}$$

Table 1 Relative Percentage Errors for Estimating Function and Derivatives Values Using CSM Directly on a 3rd degree Polynomial Function

X	F	FDX	FDXX	ERR-F	ERR-FDX	ERR-FDXX
-10.000	-626.000	223.000	-52.000	0.003	0.000	0.000
-8.947	-418.906	171.587	-45.684	0.002	0.000	0.000
-7.895	-262.431	126.823	-39.368	0.002	0.000	0.000
-6.842	-149.578	88.706	-33.053	0.002	0.000	0.000
-5.789	-73.348	57.238	-26.737	0.001	0.000	0.000
-4.737	-26.744	32.418	-20.421	0.001	0.000	0.000
-3.684	-2.766	14.247	-14.105	0.001	0.000	0.000
-2.632	5.582	2.723	-7.789	0.000	0.000	0.000
-1.579	5.299	-2.152	-1.474	0.000	0.000	0.000
-0.526	3.383	-0.380	4.842	0.000	0.000	0.000
0.526	6.833	8.042	11.158	0.001	0.000	0.000
1.579	22.646	23.111	17.474	0.001	0.000	0.000
2.632	57.820	44.828	23.789	0.001	0.000	0.000
3.684	119.354	73.194	30.105	0.002	0.000	0.000
4.737	214.245	108.208	36.421	0.002	0.000	0.000
5.789	349.492	149.870	42.737	0.002	0.000	0.000
6.842	532.093	198.180	49.053	0.002	0.000	0.000
7.895	769.046	253.138	55.368	0.003	0.000	0.000
8.947	1067.348	314.745	61.684	0.003	0.000	0.001
10.000	1434.000	383.000	68.000	0.003	0.000	0.000

Next, data values for the function were generated as pairs $\{x_i, f(x_i)\}_{i=1}^{n}$ and the data used directly to estimate the function values and its derivatives using BGRNN. Table 2 shows the results for this, using the same notation as used in Table 1. The number of data points used was n= 200 with 2 boosting steps. The increased relative errors are due to the additional errors introduced by the BGRNN function approximator. The error analysis of this approximator has been thoroughly discussed in a previous annual report [Butuk, 2006]. Except for the boundary values, it is still highly encouraging to note that for the most part the relative percentage errors are of the order of 0.1%. The excessive errors at the boundary are due to boundary effects which was also discussed in the previous annual report. In the report it was additionally discussed various techniques of correcting for boundary effects.

Table 2 Relative Percentage Errors for Estimating Function and Derivatives Values Using CSM Via BGRNN Approximator on a 3rd degree Polynomial Function

X	F	FDX	FDXX	ERR-F	ERR-FDX	ERR-FDXX
-10.00	-626.00	223.00	-52.00	0.24	32.00	900.45
-9.65	-550.42	205.00	-49.88	0.03	1.27	11.49
-9.24	-471.59	185.34	-47.46	0.00	0.05	0.57
-8.84	-400.51	166.65	-45.03	0.00	0.06	0.11
-8.43	-336.78	148.95	-42.61	0.00	0.06	0.25
-8.03	-280.01	132.22	-40.18	0.00	0.07	0.79
-7.63	-229.80	116.48	-37.76	0.00	0.05	0.70
-7.22	-185.76	101.71	-35.33	0.00	0.07	0.15
-6.82	-147.48	87.92	-32.91	0.00	0.07	0.13
-6.41	-114.57	75.12	-30.49	0.00	0.07	0.14
-6.01	-86.65	63.29	-28.06	0.00	0.07	0.16
-5.61	-63.30	52.44	-25.64	0.00	0.08	0.15
-5.20	-44.14	42.57	-23.21	0.00	0.08	0.16
-4.80	-28.77	33.68	-20.79	0.00	0.08	0.17
-4.39	-16.79	25.77	-18.36	0.00	0.09	0.18
-3.99	-7.81	18.84	-15.94	0.00	0.09	0.18
-3.59	-1.43	12.89	-13.52	0.01	0.09	0.19
-3.18	2.74	7.92	-11.09	0.00	0.09	0.18
-2.78	5.10	3.93	-8.67	0.00	0.08	0.16
-2.37	6.04	0.91	-6.24	0.00	0.03	0.02
-1.97	5.97	-1.12	-3.82	0.00	0.23	0.46
-1.57	5.27	-2.17	-1.39	0.00	0.17	0.34
-1.16	4.35	-2.24	1.03	0.00	0.15	0.35
-0.76	3.59	-1.34	3.45	0.00	0.08	0.31
-0.35	3.40	0.55	5.88	0.00	0.68	7.84
0.05	4.16	3.41	8.30	0.01	0.09	0.91
0.45	6.28	7.26	10.73	0.00	0.14	0.13
0.86	10.16	12.08	13.15	0.00	0.14	0.27
1.26	16.18	17.88	15.58	0.00	0.14	0.31
1.67	24.74	24.67	18.00	0.01	0.10	0.53
2.07	36.24	32.43	20.42	0.00	0.17	0.35
2.47	51.08	41.17	22.85	0.00	0.10	0.58
2.88	69.65	50.89	25.27	0.00	0.11	0.24
3.28	92.34	61.60	27.70	0.00	0.11	0.21
3.69	119.56	73.28	30.12	0.00	0.10	0.20
4.09	151.69	85.94	32.55	0.00	0.10	0.18
4.50	189.13	99.58	34.97	0.00	0.09	0.18
4.90	232.29	114.20	37.39	0.00	0.09	0.17
5.30	281.55	129.80	39.82	0.00	0.08	0.22
5.71	337.31	146.38	42.24	0.00	0.08	0.11
6.11	399.97	163.93	44.67	0.00	0.08	0.14
6.52	469.92	182.47	47.09	0.00	0.07	0.18
6.92	547.55	201.99	49.52	0.00	0.07	0.15
7.32	633.28	222.49	51.94	0.00	0.07	0.12
7.73	727.48	243.96	54.36	0.00	0.03	0.17
8.13	830.55	266.42	56.79	0.00	0.09	0.23
8.54	942.90	289.85	59.21	0.00	0.06	0.14
8.94	1064.92	314.27	61.64	0.00	0.06	0.02
9.34	1197.00	339.66	64.06	0.00	0.02	2.36
9.75	1339.53	366.04	66.49	0.04	1.31	316.48

Finally to be presented are the results of applying the BGRNN together with the theorem discussed above to solve a PDE problem. The PDE chosen for solution is the wave dominated cubic Schrodinger equation. The results of this application are described in the paper attached to the appendix of this report. The results of the paper will be presented at the 10th Annual Nanotechnology Conference and Trade Show - Nanotech 2007 to be held at the Santa Clara Convention Center, Santa Clara, CA., May 20-24, 2007. The paper is self explanatory and what is not fully disclosed in it has been described in this report. The complete computer program (used to solve the PDE described in paper) to be reviewed for possible intellectual property protection is attached in the appendix: Developing the boosted complex subroutine was a difficult task.

Conclusions and Recommendations

A complete PDE solver that is meshfree has been presented in this report. The solver was used to obtain compete solution of the hard to solve cubic Schrodinger equation useful in the area of nano-computing. The results are summarized in a complete paper attached in the appendix. Work is in progress to demonstrate the utility of the meshfree solver for CFD via a masters thesis. It is expected that the CFD flow equations will be easier to solve compared to the Schrodinger equations.

References

Butuk, N.K., 2006, Mathematically Reduced Chemical Reaction Mechanism Using Neural Networks, Annual Report US Department of Energy Grant Number: DE-FG26-03NT-41913
Chung, T.J., 2002, Computational Fluid Dynamics, Cambridge University Press

Kok-Lam L.,Crassidis,J.L, and Cheng, Y., 2005,"New Complex-Step Derivative Approximations with applications to second-order Kalman Filtering," AIAA paper 2005-5944, AIAA Guidance, Navigation and Control Conference, August 15-18, San Francisco, CA

Lobo, M., 1997, Time-Marching: A Step-by-Step Guide to a Flow Solver, Cranfield Series on Turbomachinery Technology, Ashgate Publishing Company.

Poinsot, T.J., and Lele, S.K., 1992, "Boundary Conditions for Direct Simulations of Compressible Viscous Flows," Journal of Computational Physics, Vol. 101, pp 104-129

Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T., 1992, Numerical Recipes in FORTRAN: The Art of Scientific Computing,Cambridge University Press

10th Annual Nanotechnology Conference and Trade Show - Nanotech 2007
Santa Clara Convention Center, Santa Clara, CA., May 20-24, 2007

A New Meshfree Accurate Method for Solving the Time Dependent Schrodinger Equation

Butuk N. K.
Prairie View A & M University
Department of Mathematics
Prairie View TX. 77446-4189
Phone: 936-261-1971
Fax: 936-857-2019
Email: nebutuk@pvamu.edu

Introduction

The current focus on developing renewable energy resources from sources such as coal will require an understanding of the complex chemistry of coal and its reactions. This understanding will have to be across all scales from nano to macro. At the nano level, to understand the chemical physics of coal will require the development of theoretical models that could replace or complement difficult, expensive and sometimes impossible experiments. The models will be developed based on studying coal's chemical reactions to establish possible reaction- pathways theoretically. Solution of the Schrodinger equation is often part of such studies. Since the Schrodinger equation can only be solved exactly in a few simple cases, numerical methods are used in practical applications.

The nano scale description of many-body systems like atoms or nuclei is based on a many-body Hamiltonian. The related wave functions are given by Slater determinants for fermions. In the case if time dependent process, such as potentials that are time varying, the situation is complex and the time-dependent Scrodinger equation (TDSE) has to be solved.

The degrees of freedom for a particular molecular system increases with the number of atoms in the system. In order to reduce this number, the Born-Oppenheimer approximation is used. This approximation is based on the great difference of masses of the elections and their nuclei in a molecule. When the nuclei move, the elections almost instantly adjust to their new positions. The time dependent Schrodinger equation is then formulated as:

$$ih\frac{\partial}{\partial t}\psi(x,t) = H(x,t)\psi(x,t)$$

Where x is a vector in a d-dimensional space representing the coordinates of the nuclei in some coordinate system. ψ(x,t) is the wave function. In one-dimensional space, the Hamiltonian operator H(x,t) is

$$H(x,t) = -\frac{h^2}{2m}\frac{\partial^2}{\partial x^2} + V_p(x) + V_c(x,t)$$

Where $V_p(x)$ represents the potential surfaces and $V_c(x,t)$ represents the external time dependent interactions.

Due to the oscillatory nature of the solutions of the time dependent Schrodinger equations (TDSE), a high resolutions in the numerical method is usually required. This implies that execution time and memory usage are very high, especially for systems with several degrees of freedom. In this paper, we introduced a meshfree method that is of high accuracy. Being meshfree, means that the solution will not be dependent on the connectivity between grid points i.e., it will not depend on the density of the grid points. This will allow the placement of the grid points adaptively as the solutions advances. Where needed, fine grids will be used and coarse grids will be used in regions of lower resolution. This adaptively speeds up the usual time consuming nature required to obtain accurate solutions. Meshfree methods also allow to easily describe both multi-scale and multi-physics problems.

The standard solutions scheme of TDSE, consists of the following two steps: first apply a suitable scheme for space discretization (usually by using some high resolutions finite difference method) and then perform the time integration. Following this approach, Finite Elements Methods (FEM) and Boundary Element Methods (BEM) have been used, Raudas and Mohan (2002). Since the TDSE, evolves with solutions containing highly oscillatory components, the time integrator requires time-steps, which are restricted by the inverse of highest frequency. Other grid discretization based techniques that have been used, are Finite Difference (FD) and the Fourier Transform (FT). These discretization schemes are implemented via the operator-splitting method, Hu et. al. (2000a). Chebyshev polynomial expansion methods as well as Wavelet based methods have also been used. The widely used scheme is the FEM. There are two problems with FEM: the method requires large storage capacity when extended to 3-dimensional problems. The second is the need to remove the singularities inherent in the nuclear potentials, Bandrouk (1993).

The TDSE is defined on the Hilbert function space, with dimension proportional to the number of atoms involved. This means that for medium size molecules, the curse of dimensionality leads to an exponentially growing computational cost of traditional grid discretization techniques based on FD, FEM or the FT. As a fix for this of curse dimensionality, two alternatives to traditional grid discretization techniques are available: sparse grids, Griebel and Zumbusch, (1998) and particle methods, Griebel and Schweitzer, (2002). Both alternatives scale reasonable well to medium dimensional problems. Sparse grids, however, are best suited for representing smooth densities with grid-aligned features i.e they are not meshfree. Particle methods or meshfree methods have also been implemented via the Radial Basis Functions (RBF) and Reproducing Kernel (RK) approach Hu et. al. (2000a & b) and found to be comparable to standard split-operator based method and the Chebyshev expansion methods for 2-Dimensional problems. The RBF and RK were found to be superior for 3-D problems. Overall, this means that for practical application meshfree methods (MFM) should be developed for TDSE.

In this paper we introduce a new meshfree method that uses a simple kernel method enhanced to a higher order of accuracy, Wand and Jones (1995). The simple kernel function approximator is combined with the complex-step method (CSM), of computing derivatives of complex functions, Butuk and Pemba, (2003). The main distinguishing feature of the method is the computation of the second-order derivatives in the TDSE, via a novel approach that is independent of grid spacing, i.e., it is truly meshfree. All that the method requires, is that the step size used be small (h<<l) and the computation of the second order derivatives is carries out in O(n) operations. The derivatives computed are accurate to $O(h^4)$. To demonstrate the method, we solve a model equation of TDSE, the scaled TDSE or the cubic Schrodinger equation:

$$iu_t + u_{xx} + q\,|u|^2\,u = 0$$

i = √-1 and therefore u(x,t) is complex. For q=1, it has the analytic solution

$$u(x,t) = \sqrt{2}e^{i(0.5x+0.75t)}\,\mathrm{sec}\,h(x-t)$$

|u| is a wave of magnitude √2 initially countered at x=0 which travels to the right at speed 1 without changing shape, i.e a soliton, Sanz-Serna and Christie (1986). The necessary boundary and initial conditions are provided by the exact solution. To advance the solution, a low order storage Runga Kutta (R-K) algorithm has been used as described next.

Runge Kutta Method

Consider an ODE of the form:

$$\frac{dy}{dx} = f(x,y)$$

The Runge-Kutta method (R-K) computes the value of f(x,y) at strategic points in the rectangle bounded by the points [x, x+h, y(x), y(x+h)] and then combines them in such a way so to increase the order of accuracy. The formula involves a weighted average of values of f(x,y) inside this domain. In order to determine a point in this rectangle, it is necessary to computer the unknown y(x+αh) where α is a coefficient between 0 and 1. The R-K scheme involves several stages or applications. As each stage is added, the order of accuracy increases by 1. Selection of the coefficient is cumbersome. These may be obtained by straightforward Taylor series expansion. More information can be obtained from numerical analysis textbooks.

The most widely used and most successful R-K is the fourth order scheme:

$$\begin{aligned}
f_1 &= f(x,y)\\
f_2 &= f(x+1/2h, y+1/2hf_1)\\
f_3 &= f(x+1/2h, y+1/2hf_2)\\
f_4 &= f(x+h, y+hf_3)\\
y(x+h) &= y(x) + h[f_1 + 2f_2 + 2f_3 + f_4]6
\end{aligned}$$

An m-stage RK scheme (which is slightly different from standard formulation) and used in this paper is of the form [Lobo, 1997]:

$$
\begin{aligned}
y_1 &= y(x) + \alpha_1 h f(y) \\
y_2 &= y(x) + \alpha_2 h f(y_1) \\
&\ldots \\
&\ldots \\
y_{m-1} &= y(x) + \alpha_{m-1} h f(y_{m-2}) \\
y(x+h) &= y_m = y(x) + \alpha_m h f(y_{m-1})
\end{aligned}
$$

This R-K method uses the classical approach and adds steps between y(x) and y(x+h). It achieves increased accuracy due to weighting more recent calculations. A novel automatic step size selection algorithm was incorporated into this scheme. The details of the algorithm will be described in a follow-up paper.

Results and Discussions

The partial differential equation (PDE) that describes a wave that moves at constant speed in the x-direction is the cubic Schrodinger equation

$$iu_t + u_{xx} + |u|^2 u = 0 \tag{1}$$

where $i = \sqrt{-1}$ is complex. The solution of this equation describes what is known as a 'soliton', which is a wave of height √2 initially centered at x=0 and traveling in the x-direction at a speed of 1 without changing shape.

Our main interest in this paper is to obtain a numerical solution of equation (1) using the complex step method (CSM) of computing both the first and second derivatives of a given analytic function. First the exact solution of (1) is given by

$$u(x,t) = \sqrt{2} e^{i(0.5x+0.75t)} \operatorname{sec} h(x-t) \tag{2}$$

which is a wave with amplitude

$$|u(x,t)| = \sqrt{2} Sech(x-t) \tag{3}$$

To obtain a numerical solution, we note that since u(x,t) is complex, it can be separated into its real and imaginary parts and therefore, can write

$$u = r + is$$

Substituting this solution into (1) we separate the single PDE into two PDEs, representing the real and imaginary parts of the Schrodinger equation

$$
\begin{aligned}
r_t + s_{xx} + s(r^2 + s^2) &= 0 \\
s_t + r_{xx} + r(r^2 + s^2) &= 0
\end{aligned}
\tag{4}
$$

Now at t=0 equation (1) is

$$u(x,0) = \sqrt{2}e^{i0.5x}Sech(x) \tag{5}$$

which for the real and imaginary parts results in

$$\begin{aligned} r(x,0) &= \sqrt{2}\cos(0.5x)Sech(x) \\ s(x,0) &= \sqrt{2}\sin(0.5x)Sech(x) \end{aligned} \tag{6}$$

These are the initial conditions which was used to solve system (4). To complete the specification of the initial value problem (IVP), the boundary conditions were needed. At -∞ and +∞ , the boundary conditions for the exact solution (2) were taken as

$$u(-\infty,t) = u(\infty,t) = 0$$

Numerically the finite grid used was $x \in [-30,70]$. Hence, the boundary conditions for the system (4) were

$$\begin{aligned} r(-30,t) &= s(-30,t) = 0 \\ r(70,t) &= s(70,t) = 0 \end{aligned} \tag{7}$$

Equations (4), with initial conditions (6) and boundary conditions (7) were solved using a time matching algorithm for $t \in [0,30]$ with the x grid interval divided into n=400 points. A novel algorithm (to be fully described in a follow-up publication) was used to obtain the second order spatial derivatives in a truly meshfree approach. The first order derivatives which were obtained by the meshfree CSM was used. For time integration, a standard 4th order Runga Kutta ODE (RK) solver was used. Stability in the time direction was maintained via slight modification of the automatic time step size control of RK to incorporate random error for each rejected or failed time step.

The results below, compares the exact solution of equation (3) $|u(x,t)|_E$ with the numerical solution $|u(x,t)|_N$ calculated as

$$|u(x,t)|_N = \sqrt{r^2 + s^2} \tag{8}$$

Figure 1 below show the exact wave solution for times, T = 0, 5, 10, 15, 20, 25, and 30. This figure clearly shows the wave nature of the solution of the cubic Schrodinger equation and as discussed above is a challenge to many numerical algorithms. The meshfree approach implemented in this paper combined an accurate kernel method (Boosted Generalized Regression Neural Network) with the CSM method to advance the solution via a modified 4th – order R-K algorithm. Due to the wave nature of solution an automatic step size control algorithm must be implemented in the R-K solution. This was done via an error perturbed methods that gave promising results and is further undergoing numerical experiments.

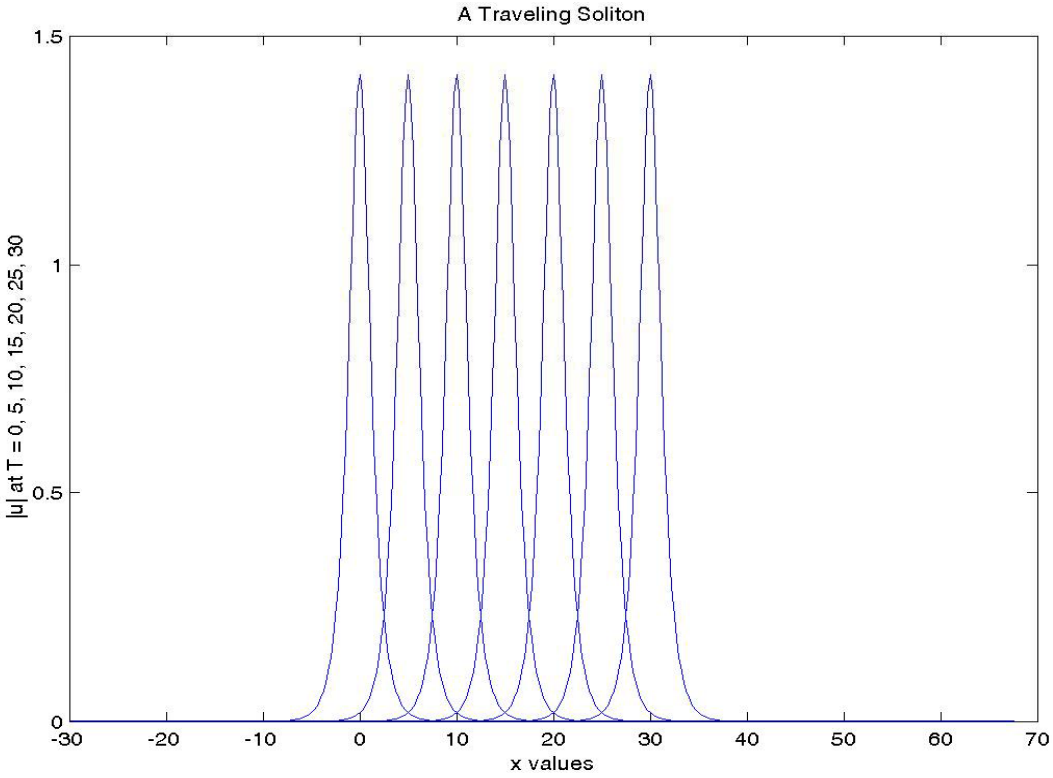

Figure 1. Exact Solution of Cubic Schrodinger Equation

Table 1, shows the numerical integration results at different time periods, T. In this table I is the grid index location (a total of 400 grid points was used), X is the grid value, the X(I)-T column shows locations where the absolute value of X(I)-T was less than 2 (only values corresponding to this are shown in the table at each T). Num, is the numerical solution, and Exact, is the exact solution. Error is the difference between the exact and numerical solution. RSS at the bottom of each time period is the sum of the squared errors shown in the table

Table 1 Numerical Integration Results at Different Time periods

I	T	X(I)	X(I)-T	Num	Exact	Error
133.00	5.01	3.08	-1.92	0.40	0.40	-0.01
134.00	5.01	3.33	-1.67	0.50	0.51	-0.01
135.00	5.01	3.58	-1.42	0.63	0.64	-0.02
136.00	5.01	3.83	-1.17	0.77	0.80	-0.03
137.00	5.01	4.09	-0.92	0.93	0.97	-0.04
138.00	5.01	4.34	-0.67	1.10	1.15	-0.05
139.00	5.01	4.59	-0.42	1.25	1.30	-0.05
140.00	5.01	4.84	-0.17	1.35	1.39	-0.04
141.00	5.01	5.09	0.08	1.38	1.41	-0.03
142.00	5.01	5.34	0.33	1.32	1.34	-0.02
143.00	5.01	5.59	0.58	1.20	1.20	-0.01
144.00	5.01	5.84	0.83	1.04	1.03	0.00
145.00	5.01	6.09	1.08	0.87	0.86	0.01
146.00	5.01	6.34	1.33	0.72	0.70	0.02
147.00	5.01	6.59	1.58	0.58	0.56	0.02
148.00	5.01	6.84	1.84	0.46	0.44	0.02
RSS	0.01					

I	T	X(I)	X(I)-T	Num	Exact	Error
153.00	10.00	8.10	-1.91	0.44	0.41	0.03
154.00	10.00	8.35	-1.66	0.55	0.52	0.03
155.00	10.00	8.60	-1.41	0.68	0.65	0.02
156.00	10.00	8.85	-1.16	0.82	0.81	0.01
157.00	10.00	9.10	-0.91	0.98	0.98	-0.01
158.00	10.00	9.35	-0.66	1.13	1.16	-0.03
159.00	10.00	9.60	-0.41	1.25	1.31	-0.05
160.00	10.00	9.85	-0.15	1.33	1.40	-0.07
161.00	10.00	10.10	0.10	1.33	1.41	-0.08
162.00	10.00	10.35	0.35	1.25	1.33	-0.08
163.00	10.00	10.60	0.60	1.13	1.19	-0.07
164.00	10.00	10.85	0.85	0.98	1.02	-0.05
165.00	10.00	11.10	1.10	0.82	0.85	-0.03
166.00	10.00	11.35	1.35	0.67	0.69	-0.02
167.00	10.00	11.60	1.60	0.54	0.55	-0.01
168.00	10.00	11.85	1.85	0.43	0.43	0.00
RSS	0.03					

I	T	X(I)	X(I)-T	Num	Exact	Error
173.00	15.00	13.11	-1.89	0.47	0.42	0.05
174.00	15.00	13.36	-1.64	0.57	0.53	0.05
175.00	15.00	13.61	-1.39	0.70	0.66	0.04
176.00	15.00	13.86	-1.14	0.84	0.82	0.02
177.00	15.00	14.11	-0.89	1.00	0.99	0.01
178.00	15.00	14.36	-0.64	1.15	1.17	-0.01
179.00	15.00	14.61	-0.39	1.27	1.31	-0.04
180.00	15.00	14.86	-0.14	1.32	1.40	-0.08
181.00	15.00	15.11	0.11	1.29	1.41	-0.11
182.00	15.00	15.36	0.36	1.20	1.33	-0.13
183.00	15.00	15.61	0.61	1.06	1.18	-0.13
184.00	15.00	15.86	0.86	0.91	1.01	-0.10
185.00	15.00	16.12	1.12	0.78	0.84	-0.06
186.00	15.00	16.37	1.37	0.65	0.68	-0.03
187.00	15.00	16.62	1.62	0.53	0.54	-0.01
188.00	15.00	16.87	1.87	0.43	0.43	0.00
RSS	0.07					

I	T	X(I)	X(I)-T	Num	Exact	Error
193.00	20.00	18.12	-1.89	0.48	0.42	0.06
194.00	20.00	18.37	-1.64	0.59	0.53	0.06
195.00	20.00	18.62	-1.38	0.71	0.67	0.05
196.00	20.00	18.87	-1.13	0.86	0.83	0.03
197.00	20.00	19.12	-0.88	1.01	1.00	0.01
198.00	20.00	19.37	-0.63	1.15	1.17	-0.03
199.00	20.00	19.62	-0.38	1.25	1.32	-0.07
200.00	20.00	19.87	-0.13	1.29	1.40	-0.11
201.00	20.00	20.13	0.12	1.26	1.40	-0.14
202.00	20.00	20.38	0.37	1.17	1.32	-0.15
203.00	20.00	20.63	0.63	1.05	1.18	-0.13
204.00	20.00	20.88	0.88	0.91	1.00	-0.09
205.00	20.00	21.13	1.13	0.79	0.83	-0.04
206.00	20.00	21.38	1.38	0.66	0.67	0.00
207.00	20.00	21.63	1.63	0.55	0.53	0.02
208.00	20.00	21.88	1.88	0.45	0.42	0.02
RSS	0.10					

I	T	X(I)	X(I)-T	Num	Exact	Error
213.00	25.00	23.14	-1.86	0.49	0.43	0.06
214.00	25.00	23.39	-1.61	0.57	0.54	0.03
215.00	25.00	23.64	-1.36	0.67	0.68	-0.01
216.00	25.00	23.89	-1.11	0.80	0.84	-0.05
217.00	25.00	24.15	-0.86	0.94	1.02	-0.08
218.00	25.00	24.40	-0.61	1.08	1.19	-0.10
219.00	25.00	24.65	-0.35	1.21	1.33	-0.12
220.00	25.00	24.90	-0.10	1.29	1.41	-0.12
221.00	25.00	25.15	0.15	1.29	1.40	-0.11
222.00	25.00	25.40	0.40	1.22	1.31	-0.09
223.00	25.00	25.65	0.65	1.11	1.16	-0.05
224.00	25.00	25.90	0.90	0.97	0.99	-0.01
225.00	25.00	26.16	1.15	0.83	0.81	0.02
226.00	25.00	26.41	1.40	0.70	0.66	0.04
227.00	25.00	26.66	1.66	0.57	0.52	0.05
228.00	25.00	26.91	1.91	0.45	0.41	0.04
RSS	0.08					

I	T	X(I)	X(I)-T	Num	Exact	Error
233.00	30.00	28.15	-1.85	0.40	0.43	-0.03
234.00	30.00	28.40	-1.60	0.52	0.55	-0.03
235.00	30.00	28.65	-1.35	0.66	0.69	-0.03
236.00	30.00	28.90	-1.10	0.80	0.85	-0.04
237.00	30.00	29.15	-0.85	0.95	1.02	-0.08
238.00	30.00	29.40	-0.60	1.07	1.19	-0.12
239.00	30.00	29.65	-0.35	1.17	1.33	-0.17
240.00	30.00	29.90	-0.10	1.23	1.41	-0.18
241.00	30.00	30.15	0.15	1.24	1.40	-0.16
242.00	30.00	30.40	0.40	1.21	1.31	-0.10
243.00	30.00	30.65	0.65	1.12	1.16	-0.03
244.00	30.00	30.90	0.91	1.00	0.98	0.02
245.00	30.00	31.15	1.16	0.85	0.81	0.04
246.00	30.00	31.40	1.41	0.71	0.65	0.05
247.00	30.00	31.65	1.66	0.57	0.52	0.05
248.00	30.00	31.90	1.91	0.47	0.41	0.06
RSS	0.13					

As seen, the accuracy of the algorithm implemented using single precision was down to order 0.1. It is expected that if the algorithm is implemented in double precision this value can be lowered further.

Acknowledgement

This work has been supported by US Department of Energy (DOE) under grant number DE-FG26-03NT-41913.

References:

1. Bandrouk, A.D, 1993, Molecules in Laser Fields, Marcel Dekker.
2. Butuk, N.K., and Pemba, J-P. ,2003,"Computing CHEMKIN Sensitivities Using Complex Variables," ASME Journal of Engineering for Gas Turbine and Power, July, pp 854-858.
3. Griebel, M., and Schweitzer, M.A., (eds), 2002, Meshfree Methods for Partial Differential Equations. Lecture Notes in Computational Science and Engineering, 26, Springer.
4. Griebel, M., and Zumbusch, G., 1998, "Adaptive sparse grids for hyperbolic conservation laws," in: Proceedings of the 7th International Conference on Hyperbolic Problems, Theory, Numeric, Applications, Birkhauser.
5. Hu, X.G., Ho, T.S., and Rabitz, H., 2000a, "Solving the bound-state Schrodinger equation by reproducing kernel interpolation," Phys. Rev. E, Vol. 61, pp 2074-2085
6. Hu, X.G., Ho, T.S., and Rabitz, H., 2000b, "Solution of the quantum fluid dynamical equation with radial basis function interpolation," Phys. Rev. E, Vol. 61, pp 5967-5976
7. Lobo, M., 1997, Time-Marching: A Step-by-Step Guide to a Flow Solver, Cranfield Series on Turbomachinery Technology, Ashgate Publishing Company.
8. Randas, L., and Mohan-R., 2002, Finite Element and Boundary Element Applications in Quantum Mechanics, Oxford University Press, London.
9. Sanz-Serna, J.M., and Christie, I., 1986, "A Simple Adaptive Technique for Nonlinear Wave Problems," Journal of Computational Physics, 67, p348-360
10. Wand, M.P., and Jones, M.C., 1995, Kernel Smoothing: Monographs On Statistics and Applied Probability 60, Chapman and Hall/CRC

ICCN – NANOSCALE MODELING

```
C     **************MAIN PROGRAM ^************************************
      PROGRAM SCHODINGER
      IMPLICIT REAL (A-H,O-Z)
      PARAMETER (NX=400) ! CHANGE IN SUBROUTINES DELY1 DELY2 CORRECTION
      DIMENSION X(NX),XX(NX),W(NX),Y3(NX),Y4(NX),CF(4),L(NX)
      DIMENSION elapsed(2),CYDX(NX),U(NX),CVDX(NX),YTEM(NX),VTEM(NX)
      COMMON/BLK1/Y1(NX),Y2(NX),YDXX(NX),V1(NX),V2(NX),VDXX(NX)
      COMMON/BLK2/YT(NX),VT(NX),DT,Q,TEMY(NX),TEMV(NX),TYV(NX)

C
C COMPLEX VARIABLES
C
      COMPLEX*8 CX(NX),CW(NX),CXX(NX),CWW(NX),CRHO


      OPEN(UNIT=78,FILE='aout2.dat',FORM='FORMATTED',
     &       STATUS='unknown')



C
C GENERATE DESIGN POINTS
C
      DATA CF/0.25,0.333,0.5,1.0/
      DATA IPInf/B'01111111100000000000000000000000'/    ! +Infinity
      DATA IMInf/B'11111111100000000000000000000000'/    ! -Infinity

      X(1)=-30.0
      X(NX)=70.0
      DX=(X(NX)-X(1))/FLOAT(NX-1)
      DO 13 I=2,NX-1
      X(I)=X(I-1)+DX
 13   CONTINUE

      RHO = DX
      Q=1.0
      ROOT2 = SQRT(2.0)
      T2 = 30.0
      PInf = transfer(IPinf,Pinf)
      Zinf = transfer(IMinf,Zinf)

C
C    COMPUTE THE FUNCTIONAL VALUES AT DESIGN POINTS: THIS IS WHERE YOU CHANGE THE FUNCTION
C    MAKE SURE CHOSEN FUNCTION MATCHES WITH (X,Y) DESIGN POINTS ABOVE
C
      DO 15 I = 1,NX
      SCH = 2.0/(EXP(X(I))+EXP(-X(I)))
      Y1(I)= ROOT2*COS(0.5*X(I))*SCH
      V1(I)= ROOT2*SIN(0.5*X(I))*SCH
      U(I) = SQRT(Y1(I)**2+V1(I)**2)
  15  CONTINUE


C
C     NEXT GENERATE SCATTERED DESIGN POINTS WITH SAME BOUNDARY
C
      W(1)=X(1)
      W(NX)=X(NX)
      W(2) = (X(1)+X(2))*0.5
      W(NX-1) =(X(NX)+X(NX-1))*0.5
      DDX=(W(NX-1)-W(2))/FLOAT(NX-3)
      DO 16 I=3,NX-2
      W(I)=W(I-1)+DDX
 16   CONTINUE


C
C     NEXT IS TO COMPUTE COMPLEX DERIVATIVES CDXY AND CDXV
C
```

```
      DO 26 I=1,NX
      CXX(I)=CMPLX(X(I))
   26 CONTINUE
      DO 28 I=1,NX
      CWW(I)=CMPLX(W(I))
   28 CONTINUE

      CRHO = CMPLX(RHO)
C
C START TIME STEPPING
C

      IC=0
      TT1  = 0.0
      TT2 = TT1
      DT = 1.0 !1.0 0.5
      DDT = DT
      EPS = 0.01 !01 0.001
  250 CONTINUE
      IC=IC+1

      RM=MOD(FLOAT(IC),2.) ! FOR CHANGING FUNCTION VALUES FOR USE IN KERNEL APPROXIMATOR (BGRNN)
      RM1=MOD(FLOAT(IC),20.) ! FOR PRINTING INTERVAL

      IF(RM.EQ.0.00) THEN
      DO 40 I = 1,NX
      CX(I) = CWW(I)
      Y1(I) = Y2(I)
      CW(I) = CXX(I)
      V1(I) = V2(I)
      XX(I) = W(I)
 40   CONTINUE

      ELSE
      DO 41 I = 1,NX
      CX(I) = CXX(I)
      CW(I) = CWW(I)
      XX(I) = X(I)
 41   CONTINUE
      ENDIF


      CALL DERIVE(NX,DX,CX,CW,Y1,Y2,CYDX,YDXX,CRHO)
      CALL DERIVE(NX,DX,CX,CW,V1,V2,CVDX,VDXX,CRHO)
C
C     TEMPORARY ARRAYS FOR R-K
C
      ICC = 0
      IDUM=10

 101  CONTINUE

      DO 30 I=1,NX
      IF(Y2(I) .EQ. PInf .OR. Y2(I) .EQ. ZInf)WRITE(*,*)'BAD DATA1' ! check for bad data
      IF(V2(I) .EQ. PInf .OR. V2(I) .EQ. ZInf)WRITE(*,*)'BAD DATA2'
      TEMY(I) = Y2(I)
      TEMV(I) = V2(I)
      TYV(I) = TEMY(I)**2 + TEMV(I)**2
  30  CONTINUE

 102  CONTINUE

C     FIRST STAGE OF R-K
      CALL DELY1
      CALL DELY2
      CALL CORRECTION (CF(1))
```

```
C     SECOND STAGE
      CALL DELY1
      CALL DELY2
      CALL CORRECTION (CF(2))

C     THIRD STAGE
      CALL DELY1
      CALL DELY2
      CALL CORRECTION (CF(3))

C     3RD STEP SOLUTION

      DO 31 I=1,NX
      YTEM(I) = TEMY(I)
      VTEM(I) = TEMV(I)
  31  CONTINUE



C     LAST STAGE
      CALL DELY1
      CALL DELY2
      CALL CORRECTION (CF(4))

C
C     TEST FOR STEP SIZE  AND NORMALIZE ERRORS IF FAILED


      DO 32 I=1,NX
      YTEM(I) = TEMY(I)- YTEM(I)
      VTEM(I) = TEMV(I)- VTEM(I)
      IF(YTEM(I).EQ.PInf .OR. YTEM(I).EQ.ZInf)YTEM(I)=1.0E3
      IF(VTEM(I).EQ.PInf .OR. VTEM(I).EQ.ZInf)VTEM(I)=1.0E3
      IF(TEMY(I).EQ.PInf .OR. TEMY(I).EQ.ZInf)TEMY(I)=1.0E3
      IF(TEMV(I).EQ.PInf .OR. TEMV(I).EQ.ZInf)TEMV(I)=1.0E3
  32  CONTINUE
      YBIG = BIG(NX,YTEM)
      VBIG = BIG(NX,VTEM)
      YVBIG = MAX(YBIG,VBIG)
      YBIGX = BIG(NX,TEMY)
      VBIGX = BIG(NX,TEMV)
      YVBIGX = MAX(YBIGX,VBIGX)
C
C     COMPUTE RELATIVE ERROR

      ERR = EPS*MAX(YVBIGX,1.0)/YVBIG

      IF(ERR .LT. 1.0) THEN

      DTEM = 0.9*DT*(ERR)**(1./4.)
      DT = SIGN(MAX(ABS(DTEM),0.1*ABS(DT)),DT)! REDUCE BY NO MORE THAN A FACTOR OF TEN

C
C     GENERATE RANDOM INDEX ARRAY
C
      DO 17 I = 1,NX
      IA = 1 + int(NX*RAN0(IDUM))
      L(I) = IA
  17  CONTINUE

      CALL NORMALIZE(NX,XMEAN,YTEM)
      CALL NORMALIZE(NX,XMEAN,VTEM)

      DO 18 I=1,NX
      TEMY(I) = Y2(I) + (1./YVBIG)*YTEM(L(I))
      TEMV(I) = V2(I) + (1./YVBIG)*VTEM(L(I))
      TYV(I) = TEMY(I)**2 + TEMV(I)**2
  18  CONTINUE
```

```
      GOTO 102

      ELSE
      TT2 = MIN(TT2 + DT, T2)
      DT = 0.9*DT*(ERR)**(1./4.)
      GOTO 200
      ENDIF


C     UPDATE

 200  IF (TT2 .GE. T2) GO TO 7000

      IF(RM.EQ.0.0) THEN
      DO 35 I=1,NX
      Y1(I) = TEMY(I)
      V1(I) = TEMV(I)
  35  CONTINUE
      ELSE
      DO 36 I=1,NX
      Y2(I) = TEMY(I)
      V2(I) = TEMV(I)
  36  CONTINUE
      ENDIF

      IF(RM1.EQ.0.00) CALL PRINT2(XX,TT2,78)
      if(tt2 .gt.5 .and. tt2.lt.5.05 .or. tt2 .gt.10 .and. tt2.lt.10.05
     +.or.tt2.gt.15 .and. tt2.lt.15.05 .or.tt2.gt.20 .and. tt2.lt.20.05
     +.or.tt2.gt.25 .and.tt2.lt.25.05 .or.tt2.gt.29.8 .and. tt2.lt.30.0
     +) CALL PRINT2(XX,TT2,78)

      GO TO 250 ! NEXT STEP
 7000 continue


C LAST STEP


      total = ETIME(elapsed)

      print *, 'End: total=', total, ' user=', elapsed(1),
     &         ' system=', elapsed(2)
      STOP
      END


      SUBROUTINE CFUNCDX(NX,X,Y1,Y2,W,RHO,HH)
      IMPLICIT COMPLEX*8 (A-H,O-Z), INTEGER(I-N)
      DIMENSION X(NX),Y1(NX),Y2(NX),W(NX)
      REAL*8 AK ! SPEED UP VARIABLES
      DO 20 K = 1,NX
      TEMP=W(K)
      W(K)=CMPLX(REAL(W(K)),REAL(HH))
      SUM1=0.0
      SUM2=0.0
          DO 30 I= 1,NX
          AK = ABS(REAL(X(I))-REAL(W(K)))
          D1 =(X(I)-W(K))**2
          H=(0.0,0.0)
      IF(AK.LE.4*REAL(RHO))
     +H =(CEXP(-(D1)/(2*(RHO)**2)))
          H1=H*Y1(I)
          SUM1=SUM1+H
          SUM2=SUM2+H1
  30      CONTINUE
      Y2(K)= SUM2/SUM1
      W(K)=TEMP
  20  CONTINUE
      RETURN
```

```
      END


      SUBROUTINE TRIDIA(A,B,C,R,U,N)
      IMPLICIT REAL (A-H,O-Z)
      PARAMETER ( NM = 500)
      DIMENSION A(N),B(N),C(N),R(N),U(N),GAM(NM)
      BET = B(1)
      IF(BET .EQ. 0)WRITE(*,*)'PIVOT ZERO RE-WRITE'
      U(1)=R(1)/BET
      DO 11 J = 2,N
      GAM(J) = C(J-1)/BET
      BET = B(J)-A(J)*GAM(J)
      IF(BET .EQ. 0)WRITE(*,*)'PIVOT ZERO STOP TRIDIA FAILTURE'
      U(J) = (R(J)-A(J)*U(J-1))/BET
 11   CONTINUE
      DO 12 J = N-1,1,-1
      U(J) = U(J) - GAM(J+1)*U(J+1)
 12   CONTINUE
      RETURN
      END


      SUBROUTINE SECONDDX(NX,DX,Y3,CYDX,U)
      IMPLICIT REAL (A-H,O-Z)
      DIMENSION Y3(NX),CYDX(NX)
      DIMENSION A(NX),B(NX),C(NX),R(NX),U(NX)
      A(1) = 0.0
      C(NX) = 0.0
      C(1) = 0.0
      A(NX) = 0.0
      R(1) = (CYDX(2)-CYDX(1))/DX
      R(NX) = (CYDX(NX)-CYDX(NX-1))/DX
C     HERE INPUT ACTUAL SECOND DERIVATIVES AT THE ENDS IF KNOWN
C     R(1) = -34.000
C     R(NX) = 50.000
      B(1)=1.0
      B(NX) = 1.0

      DO 30 I = 2,NX-1
      B(I) = 10
 30   CONTINUE
      DO 31 I = 2,NX-1
      A(I) = 1.0
      C(I) = 1.0
 31   CONTINUE
      DO 33 K = 2,NX-1
      A1 = (CYDX(K+1)+4.0*CYDX(K)+CYDX(K-1))
      A2 = (Y3(K+1)-Y3(K-1))
      HA = 3.0*A2/A1
      R(K) = (12.0/HA**2)*(Y3(K+1)-2*Y3(K)+Y3(K-1))
 33   CONTINUE

      CALL TRIDIA(A,B,C,R,U,NX)

      RETURN
      END


      SUBROUTINE DERIVE(NX,DX,CX,CW,Y1,Y2,CYDX,YDXX,CRHO)
      IMPLICIT REAL (A-H,O-Z)
      PARAMETER (NB=2,H=0.01)
      DIMENSION Y1(NX),Y2(NX),CYDX(NX),YDXX(NX)
      COMPLEX*8 CX(NX),CY1(NX),CY2(NX),CW(NX),CRHO,HX
      COMPLEX*8 CY2PT(NX),CERY(NX),CY2TM(NX),CY2PE(NX)

      HX= CMPLX(H)

      DO 25 I = 1,NX
```

```
      CY1(I)=CMPLX(Y1(I))
   25 CONTINUE

C
C     DX DERIVATIVES
C
      CALL CFUNCDX(NX,CX,CY1,CY2,CW,CRHO,HX)
      CALL CFUNCDX(NX,CX,CY1,CY2PT,CX,CRHO,HX)

C
C     START BOOSTING
C
      IC=0
   99 IC=IC+1
      DO 46 I = 1,NX
      CERY(I)=CY1(I)- CY2PT(I)
      RRR=REAL(CERY(I))
      CERY(I)=CMPLX(RRR,0.0)
   46 CONTINUE
      CALL CFUNCDX(NX,CX,CERY,CY2TM,CW,CRHO,HX)
      CALL CFUNCDX(NX,CX,CERY,CY2PE,CX,CRHO,HX)


      DO 47 I = 1,NX
      CY2PT(I)=CY2PT(I)+CY2PE(I)
   47 CONTINUE

      DO 48 J=1,NX
      CY2(J)=CY2(J)+ CY2TM(J)
   48 CONTINUE

      IF(IC.LE.NB)GO TO 99

C
C     END BOOSTING
C

      DO 30 I=1,NX
      CYDX(I)=AIMAG(CY2(I))/H
      Y2(I) = REAL(CY2(I))
   30 CONTINUE

C
C     NEXT CALCULATE THE SECOND DERIVATIVES
C
      CALL SECONDDX(NX,DX,Y2,CYDX,YDXX)

      RETURN
      END

      SUBROUTINE DELY1
      IMPLICIT REAL (A-H,O-Z)
      PARAMETER (NX=400)
      COMMON/BLK1/Y1(NX),Y2(NX),YDXX(NX),V1(NX),V2(NX),VDXX(NX)
      COMMON/BLK2/YT(NX),VT(NX),DT,Q,TEMY(NX),TEMV(NX),TYV(NX)

      YT(1) = 0.0
      YT(NX) = 0.0
      DO 30 I=2,NX-1
      YT(I) = (-VDXX(I)-Q*TYV(I)*TEMV(I))*DT
   30 CONTINUE


      RETURN
      END


      SUBROUTINE DELY2
      IMPLICIT REAL (A-H,O-Z)
```

```
      PARAMETER (NX=400)
      COMMON/BLK1/Y1(NX),Y2(NX),YDXX(NX),V1(NX),V2(NX),VDXX(NX)
      COMMON/BLK2/YT(NX),VT(NX),DT,Q,TEMY(NX),TEMV(NX),TYV(NX)

      VT(1) = 0.0
      VT(NX) = 0.0
      DO 30 I=2,NX-1
      VT(I) = (YDXX(I)+Q*TYV(I)*TEMY(I))*DT
30    CONTINUE


      RETURN
      END

      SUBROUTINE CORRECTION (CF)
      IMPLICIT REAL (A-H,O-Z)
      PARAMETER (NX=400)
      COMMON/BLK1/Y1(NX),Y2(NX),YDXX(NX),V1(NX),V2(NX),VDXX(NX)
      COMMON/BLK2/YT(NX),VT(NX),DT,Q,TEMY(NX),TEMV(NX),TYV(NX)

      DO 30 I=1,NX
      TEMY(I) = Y2(I) +(CF*YT(I))
      TEMV(I) = V2(I) +(CF*VT(I))
      TYV(I) = TEMY(I)**2 + TEMV(I)**2
30    CONTINUE


      RETURN
      END

      FUNCTION EXACT(X,T)
      IMPLICIT REAL (A-H,O-Z)
      EXACT = 2.0*(2.0/(EXP(X-T)+EXP(-(X-T))))**2
      RETURN
      END


      FUNCTION BIG(N,C)
      IMPLICIT REAL (A-H,O-Z)
      DIMENSION A(N),C(N)

      DO 11 K = 1,N
      A(K) = ABS(C(K))
11    CONTINUE

      BIG = A(1)
      DO 10 K = 2,N
      IF(BIG .LT. A(K))BIG = A(K)
10    CONTINUE
      RETURN
      END

      FUNCTION RAN0(IDUM)
      IMPLICIT REAL (A-H,O-Z)
      PARAMETER(IA=16807,IM=2147483647,AM=1./IM,
     +         IQ=127773,IR=2836,MASK=123459876)

      IDUM=IEOR(IDUM,MASK)
      K=IDUM/IQ
      IDUM=IA*(IDUM-K*IQ)-IR*K
      IF(IDUM.LT.0)IDUM=IDUM+IM
      RAN0=(AM*IDUM)
      IDUM=IEOR(IDUM,MASK)
      RETURN
      END

      SUBROUTINE NORMALIZE(N,XMEAN,A)
      IMPLICIT REAL (A-H,O-Z)
      DIMENSION A(N)
```

```
      SUM = 0.0
      DO 10 I = 1,N
      SUM = SUM + A(I)
10    CONTINUE
      XMEAN = SUM/FLOAT(N)

      DO 20 I = 1,N
      A(I)=A(I)-XMEAN
20    CONTINUE

      RETURN
      END


      SUBROUTINE PRINT2(X,T,NP)
      IMPLICIT REAL (A-H,O-Z)
      PARAMETER (NX=400)
      DIMENSION X(NX)
      COMMON/BLK2/YT(NX),VT(NX),DT,Q,TEMY(NX),TEMV(NX),TYV(NX)

      SSE = 0.0
      DO 10 I=1,NX
      IF(ABS(X(I)-T).LT.2.0)THEN
      ARG = X(I)-T
      SRN = SQRT(TYV(I))
      SRE = SQRT(EXACT(X(I),T))
      DIFF = SRN - SRE
      SSE = SSE + DIFF**2
      WRITE(NP,2) I,T,X(I),ARG,SRN,SRE,DIFF
      ELSE
      ENDIF
10    CONTINUE
      WRITE(NP,3) T,SSE
      WRITE(NP,*)
      WRITE(*,3) T,SSE

2     FORMAT(1X,I6,1X,F10.2,1X,F10.2,1X,F10.2,1X,F10.2,1X,F10.2,1X,
     +F10.2,1X,F10.2,1X,7(F10.2,1X))
3     FORMAT(1X,F10.2,1X,F10.3)

      RETURN
      END
```

Prairie View A & M University

Estimating Derivatives Using Kernel Smoothing and Complex Variables

By

Ken Everton Johnson Jr., B.S.

A THESIS

Presented to the Faculty of the Department of Mathematics

Prairie View A & M University

In Partial Fulfillment of the Requirements

For the Degree of

Master of Science in Mathematics Spring 2006

Approved as to style and Content by:

Nelson Butuk, Ph.D.
Chair of Committee

Aliakbar M. Haghighi, Ph.D.
Department Head

William H. Parker, Ph.D.
Dean, Graduate School

Arouna Davies, Ph.D.
Committee Member

Danny R. Kelley, Ph.D.
Dean, College of Arts and Sciences

ACKNOWLEDGMENT

It is indeed a privilege to have a Deity assisting me in my pursuit of this Masters of Science in Mathematics. My greatest gratitude goes out to God for placing me in the hands of Dr. Nelson Butuk, who guided me through my undergraduate studies and has been my number one mentor throughout my graduate studies. I would like to give special thanks to Mrs. Alejandrina Garza for the encouragement and other assistance she gave me during the preparation of this thesis. Special thanks to Mrs. Billie Houston who has help me since the foundation of this thesis. My appreciation goes out to my committee members Dr. Aliakbar Haghighi and Dr. Arouna Davies as well. My thanks are also extended to my parents, Ken and Dorothy Johnson, and my three brothers: Khavansky, Javid and Mickel.

Contents

1 Introduction and Literature Review …….…….. **5**

1.1 Research Objectives ……..…… 8

1.2 Thesis Organization ……..…… 9

2 The Histogram Estimator …… 10

2.1 Bin Probability Estimator of the Histogram …….…… 11

2.2 Bias, Variance and Mean Error of the Histogram ……... 11

3 Kernel Density Estimator …..……. 15

3.1 Statistics of the Kernel Density Estimator …… 21

3.1.1 Bias of Kernel Density Estimator …….. 22

3.1.2 Variance of the Estimator …….. 23

3.1.3 Mean Square Error of the estimator …… 24

3.2 Rates of Convergence of the Kernel Density Estimator . 24

4 Nadaraya Watson Estimator …… 27

4.1 Statistic of the Nadaraya Watson Estimator …….. 29

4.1.1 Bias of the Nadaraya Watson Estimator …… 30

4.1.2 Variance of the Nadaraya Watson Estimator … 33

4.1.3 Mean Square Error of the Nadaraya Watson Estimator …….. 34

4.2 Rates of Convergence of the Nadaraya Watson Estimator 34

5 Finding Derivatives by Complex Variables ……. 37

5.1 Finding Derivatives by Kernel Smoothing …….. 39

6 Problems, Results and Conclusion ….……. 41

6.1 Results for 2-D Nadaraya Watson Estimator ……. 43

6.2 Results for Complex Variable Method of finding Derivatives …….. 49

6.3 Results for finding Derivatives by Kernel Smoothing (Nadaraya Watson) and Complex Variables ……... 56

6.4 Asymptotic Analysis for Finding Derivatives by Nadaraya Watson Method …….……...…... 65

Results for Asymptotic Analysis for Finding Derivatives by Nadaraya Watson Method …… 66

6.5 Conclusion …… 69

References ……... 70

A Appendix A Background …… 73

A.1 Expected Value ……. 73

A.2 Variance……. 74

A.3 Bias …… 75

A.4 Mean Square Error …….. 76

A.5 Binomial Probability and Distribution ……... 76

A.6 Big Oh …… 78

A.7 Conditional Probability ……... 79

A.8 Covariance …… 79

B Appendix B ……. 81

Source Code DIRBYCV ……..…….. 84

Source Code HEAT …… 88

Source Code MULTIVARI …… 91